REMARKS

Claims 1-10 and 12-45 are pending in the application, claim 11 being canceled herein. Claims 1, 5, 19, 21, 31, 41, and 44 are the only independent claims.

Claims Rejections - 35 U.S.C. § 112

Claim 41 stands rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. The Examiner particularly points out that the term "the computer" in lines 16 and 17 lacks antecedent basis.

In response to the rejection of claim 41 under 35 U.S.C. § 112, second paragraph, that claim has been amended to delete the phrase "by the computer" so as to eliminate the objectionable term.

Claims Rejections - 35 U.S.C. §§ 102 and 103

Claims 1-3, 5-13, 15-18, 20, and 31-33, and 35-43 stand rejected under 35 U.S.C. § 103(a) 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,766,515 to Bitar et al. in view of U.S. Patent No. 6,260,150 to Diepstraten et al.

Claim 19 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over Bitar et al. in view of Diepstraten et al. and further in view of U.S. Patent No. 4,744,048 to Blanset et al.

Claims 44 and 45 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bitar et al. in view of Blanset et al.

The Examiner has allowed claims 21-30 and has indicated that claims 4, 14, and 34 would be allowable if rewritten in independent form to include all of the limitations of the base claim and any intervening claims.

Claim 1 Applicant has amended claim 1 to include the limitations of claim 11, which has been canceled. The amendment is to reduce the issues reviewable on appeal

and to clarify that the computer is being operated under an interpreter program to carry out the bytecode or pseudocode instructions.

Applicant maintains that claim 1 as amended distinguishes the invention over the art of record and particularly over Bitar et al. and Diepstraten et al, the references relied on by the Examiner in rejecting claims 1 and 11 under35 U.S.C. § 103(a).

Claim 1 recites a method for operating a computer comprising storing in a computer memory a plurality of ***pseudocode instructions***, at least some of the pseudocode instructions comprising a plurality of machine code instructions. Applicant respectfully traverses the rejection of claim 1 under35 U.S.C. § 103(a). Claim 1 recites a method for operating a computer comprising storing in a computer memory a plurality of ***pseudocode instructions***, at least some of the pseudocode instructions comprising a plurality of machine code instructions. For each of a plurality of tasks or jobs to be performed by the computer, a respective virtual thread of execution context data is automatically created which includes (a) a memory location of a next one of the ***pseudocode instructions*** to be executed in carrying out the respective task or job and (b) the values of any local variables required for carrying out the respective task or job, a plurality of the tasks or jobs each entailing execution of a respective one of the ***pseudocode instructions*** comprising a plurality of machine language instructions. The method of claim 1 additionally comprises processing each of the tasks or jobs in a respective series of time slices or processing slots under the control of the respective virtual thread, and in every context switch between different virtual threads, undertaking such context switch only after completed execution of a currently executing one of the pseudocode instructions. Pursuant to former claim 11, the creating of the virtual threads, the processing of said tasks or jobs in respective series of time slices or processing slots, and the undertaking of context switches all include the operating of the computer under an interpreter program.

Applicant's position is that the Bitar patent never once mentions pseudocode instructions. As recited in applicant's specification,

> The term "pseudocode" as used herein refers to computer instructions compiled for execution by an interpreter. An interpreter is a program which serves to translate into machine language pseudocode programs and to perform the indicated operations as they are translated. "Pseudocode" is unrelated to the hardware of a particular computer and requires conversion to the code used by the computer before the program can be used. Many pseudocode instructions entail the execution of multiple machine language instructions. Pseudocode is sometimes referred to as "bytecode."

In support of his conclusion that Bitar et al. disclose pseudocode or bytecode, the Examiner points only to the first paragraph of column 15 of the Bitar patent:

> In one embodiment of the systems of FIGS. 2a-2d, processors 12, 32 and 52 have an instruction set capable of executing a register load instruction during execution of a jump instruction. This may take the form of an instruction set having a jump instruction with a delay slot (as is available in the MIPS architecture) or the instruction set may be a very long instruction word (VLIW) architecture such that a load register instruction can be placed within the same VLIW instruction as the jump instruction.

Applicant respectfully fails to see how a simple jump instruction or a load register instruction can teach or suggest instructions requiring execution via the intercession of an interpreter program.

With respect to the use of an interpreter program, the Examiner points only to column 1, lines 25-35, of the Bitar reference:

> The thread abstraction described above is often referred to as a user-level thread. It is the entity that a user will create, using a threads interface, in order to express parallelism in a program. The operating system will provide a unit of scheduling, a virtual processor, to which a user-level thread will be mapped; the mapping may be performed statically, or dynamically when executing. This virtual processor will

> in turn be mapped to a physical processor by the operating system scheduler. Conceptually, it is useful to distinguish the user-level thread from the virtual processor.

This paragraph can be properly understood only with reference to the preceding paragraph in the Bitar patent:

> A thread model of program execution has proven to be a viable method for parallel execution of program code both in single and multiprocessor machines. Under the thread model, programs are partitioned (by the user or by a compiler) into a set of parallel activities. The instantiation of each activity during execution of the program code is called a thread; if the program code includes more than one thread, the program is said to be multi-threaded. By partitioning the program code into threads, it is possible to create a more easily maintainable, more easily understood and, possibly, faster program.

These paragraphs briefly explain multi-threading in conventional computers, where "programs are partitioned (by the user or by a compiler) into a set of parallel activities" the instantiation of which are called "threads." Each thread is paired with ("mapped to") a respective virtual processor, which in turn is mapped to a physical processor. Nothing here says anything about interpreter programs or the concomitant running of pseudocode instructions.

Claim 5 Claim 5 is made independent herein by incorporating the limitations of claim 1 prior to amendment of claim 1 herein.

Claim 5 is believed to be patentable over the prior art. The Examiner rejected claim 5 "for the same reasons as stated in the rejection of claim 4." However, claim 4 was indicated by the Examiner as containing allowable subject matter and claim 5 is believed to similarly contain allowable subject matter. None of the references of record, whether considered individually or collectively, either discloses or suggests a multi-threading method wherein each virtual threads includes a mutex and the method comprises setting the mutex of a selected virtual thread, subsequently modifying data in

the selected virtual thread, and thereafter resetting or releasing the mutex to enable access to the selected thread.

Claim 19 Claim 19 is made independent herein by incorporating the limitations of unamended claim 1.

Applicant respectfully traverses the rejection of claim 19 as being obvious over Bitar et al. in view of Diepstraten et al. and further in view of Blanset et al., in part because the teachings of the Blanset reference with respect to the control of objects on a video display would not lead one of ordinary skill in the art familiar with the teachings of Bitar et al. and Diepstraten et al. to control objects on a video display via threads.

Pursuant to claim 19, the tasks or jobs processed in respective series of time slices or processing slots under the control of the respective virtual threads include (a) controlling objects imaged on a computer display, each of said objects constituting a separate task or job assigned a respective one of said virtual threads, and (b) monitoring actuation of keys on a computer keyboard, each of said keys constituting a separate task or job assigned a respective one of said virtual threads.

None of the references relied on by the Examiner in rejecting claim 19, whether considered singly or in combination, teaches or suggests a method wherein objects on a display are assigned to respective virtual threads for controlling the appearance of the object on the display or wherein keyboard keys are assigned to respective threads for monitoring key actuation.

Blanset teaches the use of two screen buffers for enabling or facilitating context switching between two programs, each assigned to a respective one of the buffers. Blanset et al. solve the problem that "There is ... no ready mechanism for ensuring that the MS-DOS system or its applications, executing in non-protected mode, would not write to the screen buffer irrespective of the user's intent to the contrary." Blanset et al. provide such a mechanism:

> [T]he storage of screen data from at least one program (such as an MS-DOS program executing in a processor's non-protected mode as described above) is controlled by translation circuitry interposed between, on the one hand, the computer's address bus and, on the other hand, the screen buffer and an alternate screen buffer. The translation circuitry is operative for translating each screen buffer memory location address appearing on the address bus to the address of a corresponding memory location in an alternate screen buffer. When, for example, it is desired to have the image from a first program displayed, operation of the translation circuitry is enabled during the time that a second program has program control, thereby causing redirection of the screen data from the second program to the alternate screen buffer, even though the program for all the world "thinks" that it is writing to the screen buffer. On the other hand, when it is desired to have the image from the second program displayed, operation of the translation circuitry is disabled during the time that the second program has program control, thereby causing the screen data from the second program to be directed to the screen buffer.

One of ordinary skill in the art following the teachings of Bitar et al, Diepstraten et al. and Blanset et al. would provide plural buffers, one for each program seeking screen space, and translation circuitry "interposed between, on the one hand, the computer's address bus and, on the other hand, the screen buffer and an alternate screen buffer" where the "translation circuitry is operative for translating each screen buffer memory location address appearing on the address bus to the address of a corresponding memory location in an alternate screen buffer."

There is nothing in the teachings of Bitar et al., Diepstraten et al. and Blanset et al. to motivate one of ordinary skill in the art to use threads to control objects imaged on a computer display, where each of the objects constitutes a separate task or job assigned a respective virtual thread, as set forth in applicant's claim 19.

Furthermore, neither Bitar et al. nor Diepstraten et al. nor Blanset et al. say anything about the monitoring of keys of a keyboard. There is nothing in the teachings of

Bitar et al., Diepstraten et al. and Blanset et al. to motivate one of ordinary skill in the art to use threads to monitor actuation of keys on a computer keyboard, where each of the keys constitutes a separate task or job assigned a respective virtual thread, as recited in applicant's claim 19.

Claim 31 Independent claim 31 claims a multitasking method in a computer having an interpreter for executing a series of bytecode instructions each consisting of a multiplicity of machine code steps. The multitasking method comprises using the interpreter for each task of a plurality of tasks to be performed by the computer to define a respective virtual thread, executing, during each time slice of a series of consecutive time slices, bytecode instructions of a respective current thread selected from among the virtual threads, and executing a context switch from one of the virtual threads to another of the virtual threads only after execution of one of the bytecode instructions.

Applicant again respectfully traverses the Examiner's rejection of claim 31 under 35 U.S.C. § 103(a). Claim 31 recites a method for operating a computer comprising operating an ***interpreter*** in a computer to execute, during each time slice of a series of consecutive time slices, ***bytecode instructions*** of a respective current thread selected from among the virtual threads, and executing a context switch from one of the virtual threads to another of the virtual threads only after execution of one of the bytecode instructions. Neither the Bitar reference nor the Diepstraten reference mentions an interpreter or bytecode instructions, let alone executing a context switch from one virtual thread to another only after execution of one of the bytecode instructions.

These passage relied on by the Examiner (column 1 of the Bitar reference) with reference to interpreters briefly explains multi-threading in conventional computers,

where "programs are partitioned (by the user or by a compiler) into a set of parallel activities" the instantiation of which are called "threads." Each thread is paired with ("mapped to") a respective virtual processor, which in turn is mapped to a physical processor. Nothing in the cited passage of the Bitar reference says anything about interpreter programs or the concomitant running of pseudocode instructions.

Claim 44 According to independent claim 41, a multi-tasking computer comprises a memory storing state and context data of multiple threads or tasks and an interpreter operatively connected to the memory for executing a series of bytecode instructions each consisting of a multiplicity of machine code steps. The interpreter is programmed to define a respective virtual thread for each task to be performed by the computer, to execute, during each time slice of a series of consecutive time slices, bytecode instructions of a respective current thread selected from among the virtual threads, and to execute a context switch from one of the virtual threads to another of the virtual threads only after execution of one of the bytecode instructions.

Claim 44 distinguishes over the art relied on by the Examiner for the same reasons discussed above with reference to claim 31.

Claim 44 As set forth in claim 44, a computer method comprises running a timer of a computer to generate a series of time slices or processing slots, compiling input user source code into bytecode or pseudocode instructions each corresponding to a multiplicity of machine code instructions, and operating an interpreter of the computer to assign computing tasks to respective virtual threads, the assigning of the computing tasks to the virtual threads including identifying and storing state and context data for each of the computing tasks. The method further comprises in each of the time slices,

additionally operating the interpreter to execute selected ones of the bytecode or pseudocode instructions pursuant to the state and context data of a current one of the virtual threads. During a current one of the time slices, after the execution of each successive one of the selected bytecode or pseudocode instructions and only after such execution, the interpreter is further operated to check whether the current one of the time slices has elapsed or terminated since a commencement of execution of instructions pursuant to the current one of the virtual threads. Upon a determination of elapsing of the current one of the time slices, the interpreter is operated to perform a context switch.

According to the Examiner, the Bitar reference teaches a computer method comprising compiling input user source code into byte- or pseudocode instructions each corresponding to a multiplicity of machine code instructions (col. 5, lines 32-57), and operating an interpreter of the computer to assign computing tasks to respective (mapped) virtual threads, the assigning of the computing tasks to the virtual threads including identifying and storing state and context data for each of the computing tasks (col. 1, lines 25- 35, col. 4, lines 57-67, col. 5, lines 19-25, and col. 6, lines 9-12).

Applicant respectfully traverses the Examiner's characterization of the Bitar reference with respect to the subject matter of claim 44. Other than a single passing reference to compiler-driven scheduling, the Bitar reference ***never mentions compilers or interpreters.***

Applicant repeats and incorporates by reference herein arguments and observations submitted in the previously filed Amendment and reserves the right to reassert those arguments and observation in a subsequent appeal brief.

Conclusion

For the foregoing reasons, independent claims 1, 5, 19, 31, 41, and 44, as well as the claims dependent therefrom, are deemed to be in condition for allowance. In addition, the Examiner has allowed claim 21 and the claims dependent therefrom. An early Notice passing the application to issue is earnestly solicited.

The present amendment reduces and sharpens issues reviewable on appeal by incorporating the subject matter of claim 11 into claim 1. No new language has been incorporated into the claims. The present Amendment requires no further search inasmuch all of the subject matter has been already searched.

A Notice of Appeal and a Request for a three-month extension of time accompany this Amendment. A check in the amount of Nine-Hundred-and-Sixty Dollars ($960) is enclosed in payment of the Two-Hundred-and-Fifty Dollar ($250) fee for submitting a Notice of Appeal, the Five-Hundred-and-Ten Dollar ($510) fee for a three-month extension, and the Two-Hundred Dollar ($200) fee for submitting two new independent claims, all in small entity amounts.

Should the Examiner believe that direct contact with applicant's attorney would advance the prosecution of this application, the Examiner is invited to telephone the undersigned at the number below.

Respectfully submitted,
COLEMAN SUDOL SAPONE, P.C.

By: _____

R. Neil Sudol
Reg. No. 31,669
714 Colorado Avenue
Bridgeport, CT 066-05-1601
(203) 366-3560

Dated: February 22, 2006